See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/223417160

A Calculus for Interaction Nets Based on the Linear Chemical Abstract Machine

2 authors, including:

A calculus for interaction nets based on the linear chemical abstract machine

Shinya Sato

Himeji Dokkyo University, Faculty of Econoinformatics, 7-2-1 Kamiohno, Himeji-shi, Hyogo 670-8524, Japan

Ian Mackie

LIX, CNRS UMR 7161, École Polytechnique, 91128 Palaiseau Cedex, France

Abstract

Interaction nets are graph rewriting systems which are a generalisation of proof nets for classical linear logic. The linear chemical abstract machine (CHAM) is a term rewriting system which corresponds to classical linear logic, via the Curry-Howard isomorphism. We can obtain a textual calculus for interaction nets which is surprisingly similar to linear CHAM based on the multiplicative fragment of classical linear logic. In this paper we introduce a framework based on the linear CHAM to model interaction net reduction. We obtain a textual calculus for interaction nets that is closer to the graphical representation than previous attempts.

1 Introduction

Interaction nets are graphical rewriting systems, defined in a very similar way to term rewriting systems: they are user-defined by giving a signature and a set of rules over the signature. At the origin, interaction nets were inspired by linear logic proof nets [7], specifically the multiplicative part. Since interaction nets were introduced by Lafont in 1990 [8] there has been a wealth of theory and applications developed (see for instance [3,6,4,10,12,9] for just a sample).

In the study of the theory of interaction nets, various *textual* calculi have been proposed for interaction nets. Although these calculi destroy the graphical advantages, they do allow an easy way of writing them. In particular, they can provide a basis of a programming language [11], and they also serve to provide a language that is more familiar to develop proofs of properties. In [5] a calculus of interaction nets was proposed, based in part on the syntax that Lafont gave.

The purpose of this current paper is to investigate alternative calculi for interaction nets, with the aim of providing the following:

- a calculus which is close, with as few overheads as possible, to the graphical syntax;

- provide a first analysis of the cost of interaction net reduction, so that we can build up a cost model of interaction nets.

With respect to the first point, we see this paper as an increment on previous work. With respect to the second point, this paper is a first step towards the study of cost models for interaction nets, which, in the future, will give the ability to compare different interaction net encodings of languages in a formal setting.

The rest of this paper is structured as follows. In the next section we recall some basics on interaction nets. In Section 3 we review the linear chemical abstract machine. Section 4 is devoted to representing interaction nets in the linear chemical abstract machine. Section 5 is about cost models. We conclude the paper in Section 6.
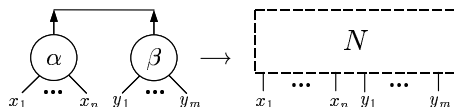
2 Interaction nets

Here we review the basic notions of interaction nets. We refer the reader to [8] for a more detailed presentation. Interaction nets are specified by the following data:

- A set Σ of *symbols*. Elements of Σ serve as *agent* (node) labels. Each symbol has an associated arity ar that determines the number of its *auxiliary ports*. If $ar(\alpha) = n$ for $\alpha \in \Sigma$, then α has $n+1$ *ports*: n auxiliary ports and a distinguished one called the *principal port*.
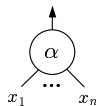
- A *net* built on Σ is an undirected graph with agents at the vertices. The edges of the net connect agents together at the ports such that there is only one edge at every port. A port which is not connected is called a *free port*.

- Two agents $(\alpha, \beta) \in \Sigma \times \Sigma$ connected via their principal ports form an *active pair* (analogous to a redex). An interaction rule $((\alpha, \beta) \to N) \in \mathcal{R}_{in}$ replaces the pair (α, β) by the net N. All the free ports are preserved during reduction, and there is at most one rule for each pair of agents. The following diagram illustrates the idea, where N is any net built from Σ.

We use the notation $N_1 \longrightarrow N_2$ for the one step reduction and $\longrightarrow^*$ for its transitive and reflexive closure.

Figure 1 shows an example of a system of interaction nets, which corresponds to the usual term rewriting system for addition on the natural numbers. Interaction nets have the following property [8]:

- **Strong Confluence**: Let N be a net. If $N \longrightarrow N_1$ and $N \longrightarrow N_2$ with $N_1 \neq N_2$, then there is a net N_3 such that $N_1 \longrightarrow N_3$ and $N_2 \longrightarrow N_3$.
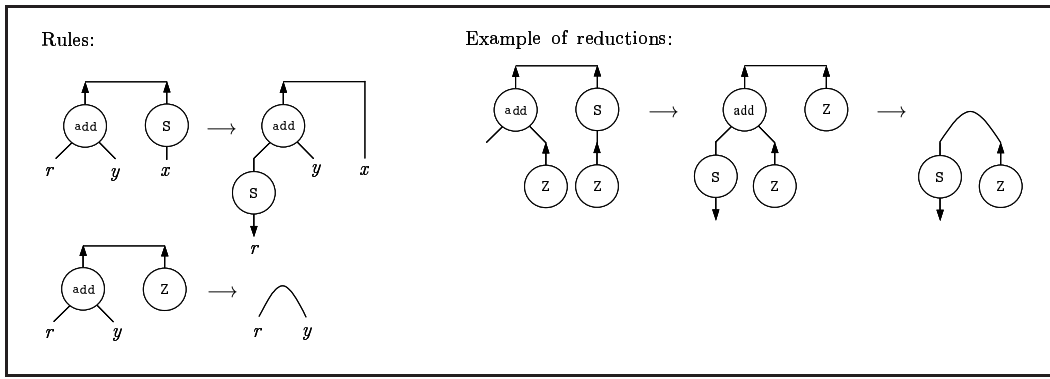
Fig. 1. An example of a system of interaction nets

3   Linear chemical abstract machine

In this section, we introduce a framework which is a generalised linear chemical abstract machine (linear CHAM), introduced by Abramsky [1]. This framework is specified by the following data:

- We assume a set $\mathcal{N}$ of *names*, ranged over by $x, y, z, ..., x_1, x_2, ...$ We use $\bar{x}, \bar{y}, ...$ to range over sequences of names.

- A set Σ of *symbols*. Elements of Σ serve as *constructors* of terms. Each symbol has an associated arity ar that determines the number of its arguments.

- *Terms* built on Σ have one of the forms: $\quad t ::= x \mid \alpha(t_1, ..., t_n)$, where $t_1, ..., t_n$ are terms, $\alpha \in \Sigma$ and $ar(\alpha) = n$. When $ar(\alpha) = 0$, the we omit brackets, i.e. we write just α. We use $t, s, u, ...$ to range over terms and $\bar{t}, \bar{s}, \bar{u}, ...$ to range over sequences of terms.

- *Coequations* have the form: $\quad t \perp s$, where t, s are terms. Coequations are molecules, thus, elements of computation. Given $\bar{t} = t_1, ..., t_k, \bar{s} = s_1, ..., s_k$, we write $\bar{t} \perp \bar{s}$ to denote the list $t_1 \perp s_1, ..., t_k \perp s_k$. We use Θ, Ξ to range over sequences of coequations.

- *Configurations* have the form: $\quad \Theta; \bar{t}$, where Θ is a sequence of coequations and $\bar{t}$ is a sequence of terms. In a configuration $\Theta; \bar{t}$, we call Θ a *solution* and $\bar{t}$ the *main body*. Main bodies are used for recording the result of the computation. We use P, Q to range over configurations.

- *Rewriting rules* have the form: $\quad \alpha(t_1, ..., t_n) \perp \beta(s_1, ..., s_k) \longrightarrow \Theta$, where $\alpha, \beta \in \Sigma$, $t_1, ..., t_n, s_1, ..., s_k$ are meta-variables for terms and Θ is a sequence of coequations. Moreover, $t_1, ..., t_n, s_1, ..., s_k$ must be distinct, occur once in Θ respectively and there should be at most one rule between α and β. We use $\mathcal{R}_{cham}$ for a set of these rules.

Definition 3.1 (free and bound) When a name x occurs twice in a term t, then the occurrences are said to be *bound* for t and all other occurrences are *free* for t. When every name occurs twice in a term t, then t is said to be *linear*. We extend this notion into name occurrences in sequences of terms, coequations and so on.

Linear CHAM is a rewriting system for configurations. The rules of linear CHAM are divided into three kinds: *structural rules* which describe the "magical mix-

3

Notation:

(1) Given Θ, we write Θ^l to denote the results of replacing each occurrence of a bounded name x for Θ by fresh names x^l respectively.

Structural Rules:

- Basic Rules:
  $t\bot u \leftrightharpoons u\bot t,$
  $t_1\bot u_1, t_2\bot u_2 \leftrightharpoons t_2\bot u_2, t_1\bot u_1$

- Structural Context Rule:
  Basic rules can be applied in any context of solutions:

  $$\frac{\Theta \leftrightharpoons \Xi}{C[\Theta]; \bar{t} \leftrightharpoons C[\Xi]; \bar{t}}$$

- Magical Mixing Rule:

  $$\frac{P_1 \leftrightharpoons^* P_2 \quad P_2 \longrightarrow Q_2 \quad Q_2 \leftrightharpoons^* Q_1}{P_1 \longrightarrow Q_1}$$

Reaction Rules:

- Communication:
  $t\bot x, x\bot u \longrightarrow t\bot u$

- Binding:
  $t\bot x, s\bot u \longrightarrow s\bot u[t/x]$
     (s, u are not names and x occurs in u)

- Change:
  $\alpha(\bar{t})\bot\beta(\bar{s}) \longrightarrow \Theta^l$
          $(\alpha(\bar{t})\bot\beta(\bar{s}) \longrightarrow \Theta \ \in \mathcal{R}_{cham})$

- Reaction Context Rule:

  $$\frac{\Theta \longrightarrow \Xi}{\Theta_1, \Theta, \Theta_2; \bar{t} \longrightarrow \Theta_1, \Xi, \Theta_2; \bar{t}}$$

Cleanup Rule:

- Cleanup: $x\bot t, \Theta; \bar{t} \longrightarrow \Theta; \bar{t}[t/x]$
                                          $(x$ occurs in $\bar{t})$

Fig. 2. Rules of Linear CHAM

ing" [2], *reaction rules* which describe the actual computation steps and a *cleanup rule* which records the result of the computation in the main body. In Figure 2 we give the structural rules, the reaction rules and a cleanup rule. We define $P \Downarrow Q$ by $P \longrightarrow^* Q$ where Q is a $\longrightarrow$-normal form.

Example 3.2 In this framework the rules of Figure 1 can be represented as the following two rules:

- $\mathtt{add}(t_1, t_2)\bot\mathtt{S}(u) \longrightarrow t_1\bot\mathtt{S}(x), \ \mathtt{add}(x, t_2)\bot u,$

- $\mathtt{add}(t_1, t_2)\bot\mathtt{Z} \longrightarrow t_1\bot t_2$

and the computation can be performed as follows:

$$\mathtt{add}(y, \mathtt{Z})\bot\mathtt{S}(\mathtt{Z}); y \ \longrightarrow \ y\bot\mathtt{S}(x), \mathtt{add}(x, \mathtt{Z})\bot\mathtt{Z}; y \ \longrightarrow$$

$$y\bot\mathtt{S}(x), x\bot\mathtt{Z}; y \ \longrightarrow \ x\bot\mathtt{Z}; \mathtt{S}(x) \ \longrightarrow \ ; \mathtt{S}(\mathtt{Z}).$$

From now on, we identify N_1 and N_2 such that $N_1 \equiv N_2$ which is defined as follows:

Definition 3.3 (structural equivalence) A *renaming* is a permutation $\rho : \mathcal{N} \cong \mathcal{N}$. This is extended to a substitution on terms, coequations, configurations etc. in the usual way. Now we define *structural equivalence* of configurations $P \equiv Q$ by: $\exists \rho.(\rho(P) \leftrightharpoons^* Q)$.

This system has the following properties, using the same ideas as [1].

Theorem 3.4 *Let a rule set $\mathcal{R}_{cham}$ and a proof expression P be linear.*

- **Linearity:** *If $P \longrightarrow Q$, then Q is also linear.*

4

- **Strong Confluency:** *If* $P \longrightarrow Q_1, P \longrightarrow Q_2$ *and* $Q_1 \not\equiv Q_2$, *then for some* R, $Q_1 \longrightarrow R$ *and* $Q_2 \longrightarrow R$.

- **Determinacy:** *If* $P \Downarrow Q_1$ *and* $P \Downarrow Q_2$, *then* $Q_1 \equiv Q_2$.

4 From interaction nets to linear CHAM

In this section we introduce a translation from interaction nets into the linear chemical framework. First, we define a translation $\mathbf{T}_N$ from nets into configurations, where we restrict nets to be deadlock free (consequently, there are no vicious circles).

- **Free ports:** For every free port, we connect a principal port of a fresh agent whose arity is 0. We use $\mathrm{T}, \mathrm{S}, \mathrm{U}$,... to range over these fresh agents.

- **Agents:** For every agent whose symbol is α, we introduce a term $\alpha(x_1, ..., x_n)$ where each $x_1, ..., x_n$ is a fresh name. In this translation, the occurrence of the term $\alpha(x_1, ..., x_n)$ is corresponding to the principal port of the agent α, and each occurrence of $x_1, ..., x_n$ corresponds to the auxiliary ports respectively.

- **Connections between principal ports:** We assume that terms for these principal ports are $\alpha(\bar{t})$ and $\beta(\bar{s})$. For this connection, we introduce a coequation $\alpha(\bar{t}) \perp \beta(\bar{s})$.

- **Connections between a principal port and an auxiliary port:** We assume that terms for a principal port and an auxiliary port are $\alpha(\bar{t})$ and x respectively. For this connection, we replace the occurrence of x by $\alpha(\bar{t})$.

- **Connections between auxiliary ports:** We assume that names for these auxiliary ports are x and y respectively. For this connection, we introduce a fresh name z and we replace the occurrence of x and y by z.

- **Finalization:** We make a configuration in the following way:
  · for the solution, collect coequations generated by this translation.
  · the main body must be empty because all free principal ports are connected to principal ports of fresh agents.

For example, each net in the example of reductions in Figure 1 is represented as follows:

$$\mathtt{add}(\mathrm{T}, \mathrm{Z}) \perp \mathtt{S}(\mathrm{Z}); , \qquad \mathrm{T} \perp \mathtt{S}(x), \mathtt{add}(x, \mathrm{Z}) \perp \mathrm{Z}; , \qquad \mathrm{T} \perp \mathtt{S}(\mathrm{Z}); .$$

We can show that all coequations obtained by using $\mathbf{T}_N$ have the form $t \perp u$ where t and u are not names because every occurrence of a coequation is caused as a result of a connection between principal ports:

Lemma 4.1 *When N has no vicious circle, then all coequations obtained by $\mathbf{T}_N[N]$ have forms $t \perp u$ where t and u are not names.* $\qquad\square$

Next we define a translation $\mathbf{T}_R$ of a rule $((\alpha, \beta) \to N) \in \mathcal{R}_{in}$.

- **The left hand side:** We introduce a coequation $\alpha(t_1, ..., t_n) \perp \beta(s_1, ..., s_k)$.

- **The right hand side:** Assuming each auxiliary port in the right hand side are connected to principal ports $t_1, ..., t_n, s_1, ..., s_k$ corresponding to the occurrences of $\alpha(t_1, ..., t_n) \perp \beta(s_1, ..., s_k)$ respectively, we have a sequence of coequations Θ by

using the translation $\mathbf{T}_N$ of the nets of the right hand side.

- **Finalization:** We make a rule

$$\alpha(t_1, ..., t_n)\perp\beta(s_1, ..., s_k) \longrightarrow \Theta(t_1, ..., t_n, s_1, ..., s_k),$$

where $\Theta(t_1, ..., t_n, s_1, ..., s_k)$ means meta variables $t_1, ..., t_n, s_1, ...s_k$ occur in Θ respectively.

For example, each rule in Figure 1 is represented as follows:

- $\mathtt{add}(t_1, t_2)\perp\mathtt{S}(s_1) \longrightarrow t_1\perp\mathtt{S}(x), \ \mathtt{add}(x, t_2)\perp s_1,$
- $\mathtt{add}(t_1, t_2)\perp\mathtt{Z} \longrightarrow t_1\perp t_2.$

4.1 Correctness

Next, we examine a correctness of the translation. In Figure 1, the first net in the example of reductions is reduced to the last net by using rules obtained from the translation $\mathbf{T}$ as follows:

$$\mathtt{add}(\mathrm{T},\mathrm{Z})\perp\mathtt{S}(\mathrm{Z}); \ \longrightarrow \ \mathrm{T}\perp\mathtt{S}(x), \mathtt{add}(x,\mathrm{Z})\perp\mathrm{Z}; \ \longrightarrow \ \mathrm{T}\perp\mathtt{S}(x), x\perp\mathrm{Z}; \ \longrightarrow \ \mathrm{T}\perp\mathtt{S}(\mathrm{Z}); .$$

This correspondence shows the correctness of the this translation for the case of Figure 1. We can show the correctness for the other cases:
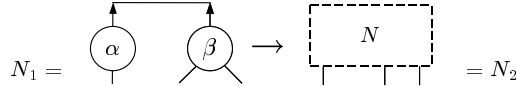
Theorem 4.2 *Let $\mathcal{R}_{in}$ and N_1 be linear. When $N_1 \longrightarrow N_2$ by using a rule $R \in \mathcal{R}_{in}$ and N_1, N_2 does not contain any vicious circles, then by using a rule $\mathbf{T}_R[R]$, either $\mathbf{T}_N[N_1] \longrightarrow \mathbf{T}_N[N_2]$ or $\mathbf{T}_N[N_1] \longrightarrow \cdot \longrightarrow_{Binding} \mathbf{T}_N[N_2]$.*

Proof. We argue just a simple case such that $((\alpha, \beta) \to N) \in \mathcal{R}_{in}$ where $ar(\alpha) = 1$, $ar(\beta) = 2$, and $\mathbf{T}_R[(\alpha, \beta) \to N] = \alpha(t)\perp\beta(s_1, s_2) \longrightarrow \Theta(t, s_1, s_2)$.
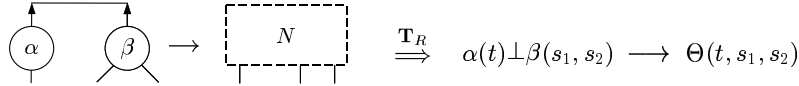
We check connections of free ports of (α, β):

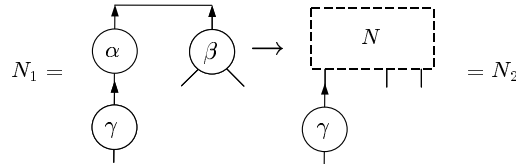**No port is connected:** N_1 is just a net (α, β).

Then, by the construction of $\Theta(t, s_1, s_2)$,
$\mathbf{T}_N[N_1] = \alpha(\mathrm{T})\perp\beta(\mathrm{S}_1, \mathrm{S}_2); \ \longrightarrow \ \Theta(\mathrm{T}, \mathrm{S}_1, \mathrm{S}_2); \ = \mathbf{T}_N[N_2].$

One port is connected: We assume that the most left free port is connected.
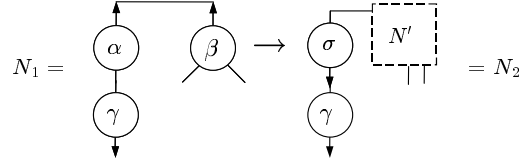 **To a principal port:** A principal port of an agent γ is connected as follows:

Because the principal port of γ is represented as a term, we can show
$\mathbf{T}_N[N_1] = \alpha(\gamma(\mathrm{T}))\perp\beta(\mathrm{S}_1, \mathrm{S}_2); \ \longrightarrow \ \Theta(\gamma(\mathrm{T}), \mathrm{S}_1, \mathrm{S}_2); \ = \mathbf{T}_N[N_2].$
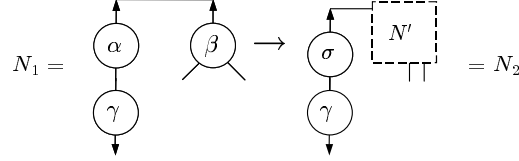 **To an one auxiliary port:** An auxiliary port of an agent γ is connected.

6

Case 1: In the right hand side of the rule, the auxiliary port of γ is connected to a principal port:

$$N_1 = \qquad\qquad \rightarrow \qquad\qquad = N_2$$

Let $\mathbf{T}_R[(\alpha, \beta) \to N] = \alpha(t) \perp \beta(s_1, s_2) \longrightarrow t \perp \sigma(u), \Theta'(s_1, s_2)$.
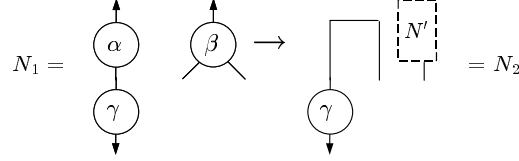Then, $\mathbf{T}_N[N_1] = \mathrm{T} \perp \gamma(x), \alpha(x) \perp \beta(\mathrm{S}_1, \mathrm{S}_2); \longrightarrow \mathrm{T} \perp \gamma(x), x \perp \sigma(u), \Theta'(\mathrm{S}_1, \mathrm{S}_2);$
$\qquad \longrightarrow_{Binding} \mathrm{T} \perp \gamma(\sigma(u)), \Theta'(\mathrm{S}_1, \mathrm{S}_2); = \mathbf{T}_N[N_2]$.

Case 2: In the right hand side of the rule, the auxiliary port of γ is connected to an auxiliary port:

$$N_1 = \qquad\qquad \rightarrow \qquad\qquad = N_2$$

Then, $\mathbf{T}_N[N_1] = \mathrm{T} \perp \gamma(x), \alpha(x) \perp \beta(\mathrm{S}_1, \mathrm{S}_2); \longrightarrow \mathrm{T} \perp \gamma(x), \Theta(x, \mathrm{S}_1, \mathrm{S}_2);$
$\qquad = \mathbf{T}_N[N_2]$.

Case 3: In the right hand side of the rule, the auxiliary port of γ is connected to a free port.

$$N_1 = \qquad\qquad \rightarrow \qquad\qquad = N_2$$

Let $\mathbf{T}_R[(\alpha, \beta) \to N] = \alpha(t) \perp \beta(s_1, s_2) \longrightarrow t \perp s_1, \Theta''(s_2)$.
Then, $\mathbf{T}_N[N_1] = \mathrm{T} \perp \gamma(x), \alpha(x) \perp \beta(\mathrm{S}); \longrightarrow \mathrm{T} \perp \gamma(x), x \perp \mathrm{S}_1, \Theta''(\mathrm{S}_2);$
$\qquad \longrightarrow_{Binding} \mathrm{T} \perp \gamma(\mathrm{S}_1), \Theta''(\mathrm{S}_2); = \mathbf{T}_N[N_2]$.

Two ports are connected: In the case that these ports are connected to distinct agents respectively, we can show the correspondence as same as above. Next, we check the case that these ports are connected to the same agent:

$$N_1 = \qquad\qquad \rightarrow \qquad\qquad = N_2$$
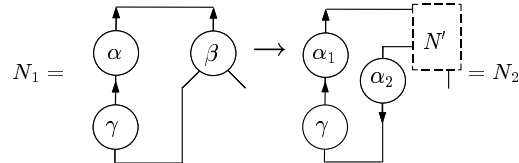
Let $\mathbf{T}_R[(\alpha, \beta) \to N] = \alpha(t) \perp \beta(s_1, s_2) \longrightarrow s_1 \perp \alpha_2(u), \Theta'(t, s_2)$.
Then, $\mathbf{T}_N[N_1] = \alpha(\gamma(x)) \perp \beta(x, \mathrm{S}_2); \longrightarrow x \perp \alpha_2(u), \Theta'(\gamma(x), \mathrm{S}_2);$
$\qquad \longrightarrow_{Binding} \Theta'(\gamma(\alpha_2(u)), \mathrm{S}_2); \qquad \text{(by using the Lemma 4.1)}$
$\qquad = \mathbf{T}_N[N_2]$.

The other cases are also shown in a similar way. We note that nets such as the following are not treated because there exists some vicious circle in N_2:

$$N_1 = \qquad \longrightarrow \qquad = N_2$$

□
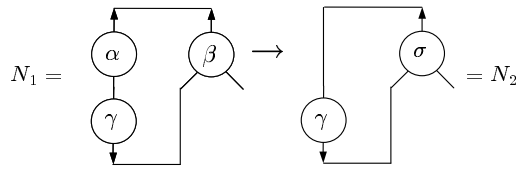
## 4.2 How far from computation of interaction nets

Here we ask how close our representation of interaction nets are to the graphical ones. The number of reductions may be increased more than the number of reductions in Interaction Nets. For example, in Figure 1, the number of reductions is two, but in the linear chemical framework, the total number of reductions is three, even though the number of reductions by using the change rule is two.

This is because we have to introduce names for connections between auxiliary ports. Therefore, reductions for these names can be increased according to the number of these connections at most. We introduce a weight for the number which can be increased:

- For a net N, a weight $|N|$ is the number of connections between auxiliary ports.
- For a rule $((\alpha, \beta) \to N) \in \mathcal{R}_{in}$, a weight $|(\alpha, \beta) \to N|$ is $|N|$.

When $N_1 \Downarrow N_2$, the sum of $|N_1|$ and $|r|$ for each rule r which is applied in order to obtain N_2 is the number of reductions which can be increased in the case of the translation **T**. As an example, in Figure 1, in the net of the right hand side in the rule between add and S, one name is introduced. Therefore the number of reductions is increased by one.

On the other hand, this redundancy is needed for parallel computation. For example, in the following net, there are two active pairs whose auxiliary ports are connected together:
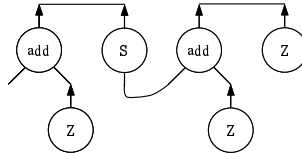
When these reductions are performed simultaneously, there becomes no information about the connection between these auxiliary ports. For this reason, we need some device for preserving this information. In the case of our calculus, this net is represented as $\mathtt{add(T, Z) \bot S(x), add(x, Z) \bot Z;}$, then, after these reductions that can be done simultaneously, we can rewire these auxiliary ports together. Therefore, we can think that this redundancy is the price that we have to pay for parallel computations.

4.3 Related works

In [5] a textual calculus for interaction nets has been proposed. The main difference between this calculus and our calculus is a way of representation of rewriting rules. In the textual calculus, fresh names are introduced according to occurrences

of names in a rule, even if the occurrences are not for connections between auxiliary ports. Therefore, the number of reductions can be increased according to the number of names which is not for connections between auxiliary ports in rules at most.

For example, the rules in Figure 1 are defined as follows:

$$\mathtt{add}(\mathtt{S}(x), y) \bowtie \mathtt{S}(\mathtt{add}(x, y)) \qquad \mathtt{add}(x, x) \bowtie \mathtt{Z}.$$

Note that, in these rules, the number of names which are not for connections between auxiliary ports are one respectively. The rewritings are performed as follows:

$$\langle\, a \mid \mathtt{add}(a, \mathtt{Z}) = \mathtt{S}(\mathtt{Z}) \,\rangle \longrightarrow \langle\, a \mid a = \mathtt{S}(x'), \mathtt{Z} = y', \mathtt{Z} = \mathtt{add}(x', y') \,\rangle$$
$$\longrightarrow \langle\, \mathtt{S}(x') \mid \mathtt{Z} = y', \mathtt{Z} = \mathtt{add}(x', y') \,\rangle \longrightarrow \langle\, \mathtt{S}(x') \mid \mathtt{Z} = \mathtt{add}(x', \mathtt{Z}) \,\rangle$$
$$\longrightarrow \langle\, \mathtt{S}(x') \mid x' = x'', \mathtt{Z} = x'' \,\rangle \longrightarrow \langle\, \mathtt{S}(x'') \mid \mathtt{Z} = x'' \,\rangle \longrightarrow \langle\, \mathtt{S}(\mathtt{Z}) \mid \,\rangle.$$

Compared with the number of reductions in our calculus, we can find that the number of reductions is increased by two (see Example 3.2).

5 Cost models

In this section we briefly outline how we propose to measure the cost of an interaction net computation using the calculus. The main point is that we can use the calculus to give a precise measure.

In the graphical notation, for the rule between $\mathtt{add}$ and $\mathtt{S}$ in Figure 1, we may estimate that the cost should be three times of the cost of a rewire of ports intuitively:
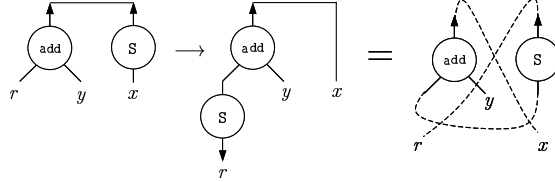
However, each cost of rewires are different. For example, as a result of a rewire, whether a new active pair can be created or not depends on kinds of connections. Therefore, it is difficult to estimate the cost.

In our calculus, we can divide rewires into two sorts: one is creation of an active pair, another is just a re-connection. Taking account of agents, we introduce the following constants of costs:

- $\alpha \cdots$ the cost of creation of an agent (and a name),
- $\epsilon \cdots$ the cost of erasing of an agent (and a name),
- $\theta \cdots$ the cost of creation of an active pair,
- $\omega \cdots$ the cost of rewire.

and we estimate costs of a rule as the sum of creation costs of the right hand side of the rule. As an example, for the rule $\mathtt{add}(t_1, t_2) \bot \mathtt{S}(u) \longrightarrow t_1 \bot \mathtt{S}(x),\ \mathtt{add}(x, t_2) \bot u$, the cost for the first coequation $t_1 \bot \mathtt{S}(x)$ can be estimated as the sum of the cost of one creation of a name x, one rewire of x to $\mathtt{S}$ and two rewires of both t_1 and $\mathtt{S}(x)$ to a coequation cell $\bot$, thus $\alpha + 3\omega + \theta$. For the next coequation $\mathtt{add}(x, t_2) \bot s_1$,

9

the cost can be the sum of the cost of one rewires of x to `add`, one creation of a coequation cell $\bot$ and two rewires of both $\mathtt{add}(x, t_2)$ and s_1 to the coequation cell $\bot$, thus $3\omega + \theta$. Therefore, the cost can be $\alpha + 6\omega + 2\theta$. For the another rule $\mathtt{add}(t_1, t_2) \bot \mathtt{Z} \longrightarrow t_1 \bot t_2$, the cost can be estimated as the sum of the cost of eliminations of agent `add` and `Z`, one creation of a coequation cell $\bot$ and two rewires of both t_1 and t_2 to the coequation cell $\bot$, thus $2\epsilon + \theta + 2\omega$. If we can reuse a coequation cell in the left hand side, these costs can be reduced to $\alpha + 6\omega + \theta$ and $2\epsilon + 2\omega$ respectively.

This cost model is suitable for the calculation models that treat an active pairs as a data object of two cells and store each information of connections between auxiliary ports into a buffer.

6 Conclusions

In this paper we have given a new calculus for interaction nets that we believe is closer to the graphical framework than extant calculi. We have also made a first attempt to study the cost of an interaction net computation. Current work is now devoted to building programming languages around this framework, and also using the framework to show properties of existing systems of interaction.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[2] G. Berry and G. Boudol. The chemical abstract machine. In *Conference Record of the Seventeenth Annual Symposium on Principles of Programming Languages*, pages 81–94, San Francisco, California, Jan. 1990.

[3] M. Fernández and I. Mackie. Coinductive techniques for operational equivalence of interaction nets. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS'98)*, pages 321–332. IEEE Computer Society Press, June 1998.

[4] M. Fernández and I. Mackie. Interaction nets and term rewriting systems. *Theoretical Computer Science*, 190(1):3–39, January 1998.

[5] M. Fernández and I. Mackie. A calculus for interaction nets. In G. Nadathur, editor, *Proceedings of the International Conference on Principles and Practice of Declarative Programming (PPDP'99)*, volume 1702 of *Lecture Notes in Computer Science*, pages 170–187. Springer-Verlag, September 1999.

[6] M. Fernández and I. Mackie. A theory of operational equivalence for interaction nets. In G. Gonnet, D. Panario, and A. Viola, editors, *LATIN 2000. Theoretical Informatics. Proceedings of the 4th Latin American Symposium, Punta del Este, Uruguay*, volume 1776 of *Lecture Notes in Computer Science*, pages 447–456. Springer-Verlag, April 2000.

[7] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[8] Y. Lafont. Interaction nets. In *Seventeenth Annual Symposium on Principles of Programming Languages*, pages 95–108, San Francisco, California, 1990. ACM Press.

[9] S. Lippi. λ-calculus left reduction with interaction nets. *Mathematical Structures in Computer Science*, 12(6), 2002.

[10] I. Mackie. Interaction nets for linear logic. *Theoretical Computer Science*, 247(1):83–140, September 2000.

[11] I. Mackie. Towards a programming language for interaction nets. *Electronic Notes in Theoretical Computer Science*, 127(5):133–151, May 2005.

[12] J. S. Pinto. Sequential and concurrent abstract machines for interaction nets. In J. Tiuryn, editor, *Proceedings of Foundations of Software Science and Computation Structures (FOSSACS)*, volume 1784 of *Lecture Notes in Computer Science*, pages 267–282. Springer-Verlag, 2000.